.NET Full Stack Development Program

—

Day 12 Web Application & .NET History

Outline

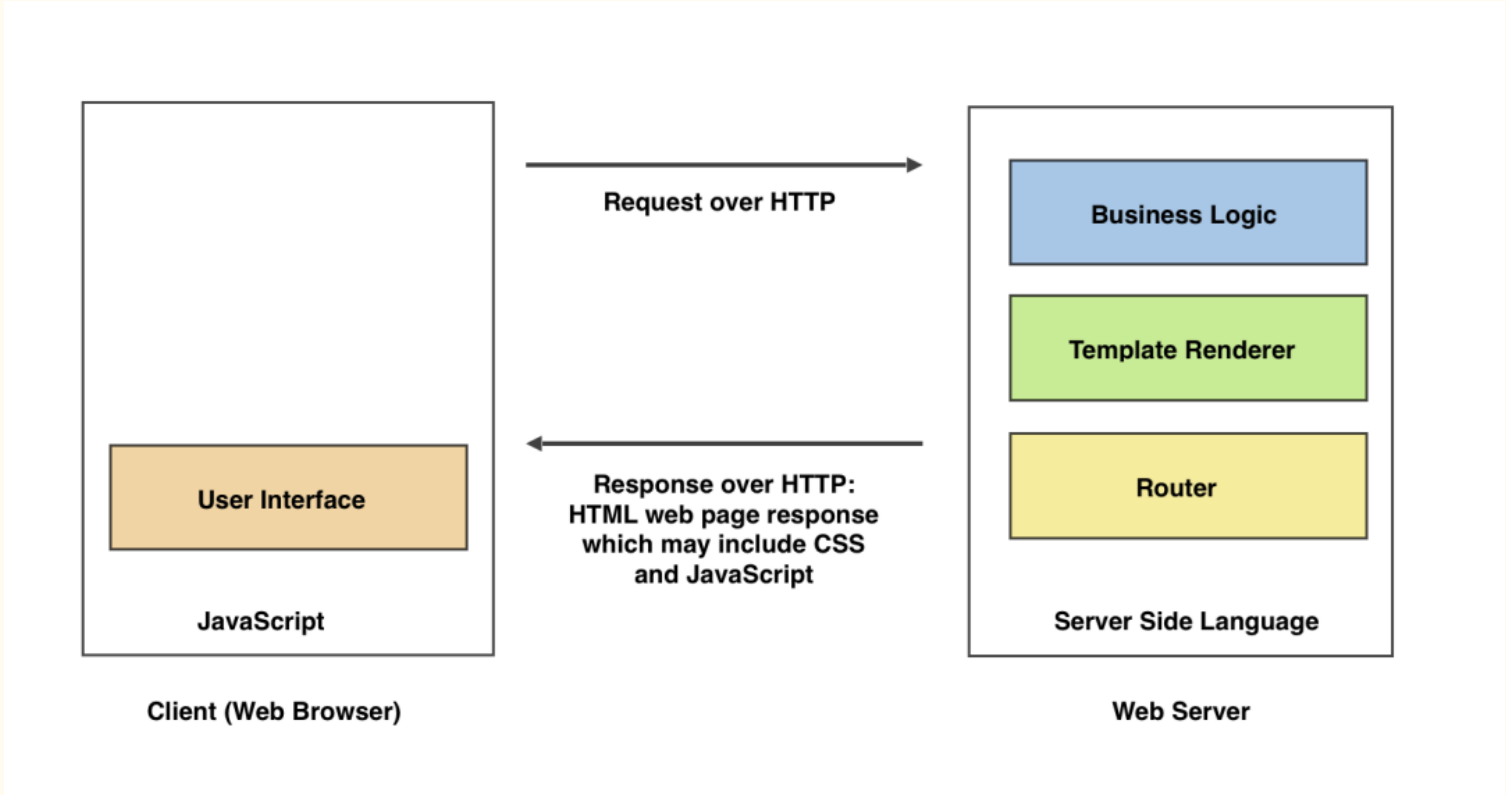
- Web Application
- HTTP Request
  - http method: get, post, put, update, delete
- .NET History
  - .NET vs. .NET Core
    - ASP.NET vs. ASP.NET Core
  - ASP.NET Core project type
    - Mvc, api, blazor, razor, ...

Web Application

- A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet.

- Web applications use a combination of server-side scripts (like .NET/Java/C++) to handle the storage and retrieval of the information, and client-side scripts (JavaScript and HTML) to present information to users

Web Application



HTTP

- HTTP stands for Hypertext Transfer Protocol and is used to structure requests and responses over the internet.

- HTTP requires data to be transferred from one point to another over the network

- http://www.google.com/

HTTPS

- Hypertext Transfer Protocol Secure, it's a secure version of HTTP
  - It's encrypted to increase the security of data transferring

- HTTPS uses TLS/SSL (Transport Layer Security/Secure Socket Layer) to encrypt communication by using an asymmetric public key infrastructure
  - Private Key – Controlled by the owner of a website
  - Public Key – Available to everyone who wants to interact with the server

- https://www.google.com/

HTTP Request

HTTP Request is a packet of Information that one computer sends to another computer to communicate something

To its core, HTTP Request is a **packet of binary data** sent by the Client to server.

An HTTP Request contains following parts

- Request Line
- Headers, 0 or more Headers in the request
- An optional Body of the Request

Request Line

A **Request Line** specifies the Method Token (**GET, PUT** ...) followed by the Request URI and then the HTTP Protocol that is being used

HTTP Request Method

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource.

- GET — The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
- POST — The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT — The PUT method replaces all current representations of the target resource with the request payload.
- DELETE — The DELETE method deletes the specified resource.
- HEAD, CONNECT, OPTIONS, TRACE, PATCH

POST VS. PUT

POST — The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line

- In other word, POST is used to create/add

PUT — The PUT method requests that the enclosed entity be stored under the supplied RequestURI.

- If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server.
- If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI.
- In other word, PUT is used to create/add and update

POST VS. PUT

- When to use
  - Do you name your URL objects you create explicitly, or let the server decide? If you name them then use PUT. If you let the server decide then use POST.
  - PUT is **idempotent**, so if you PUT an object twice, it has no effect.
  - You can update or create a resource with PUT with the same object URL
  - POST twice with the same data means that create two identical users with different ids

POST VS. PUT

Pragmatic Advice

- To save an existing user, or one where the **client generates the id,** and it's been verified that the id is unique
  - PUT /user/12345 HTTP/1.1 <-- create the user providing the id 12345
  - GET /user/12345 HTTP/1.1 <-- return that user
- Otherwise, use POST to initially create the object, and PUT to update the object:
  - POST /user HTTP/1.1 <--- create the user, server returns 12345
  - PUT /user/12345 HTTP/1.1 <--- update the user

When it comes to system design, we only need to choose one either POST or PUT and support it well in the application.

URL VS. URI

URL — uniform resource locator

URI — uniform resource identifier

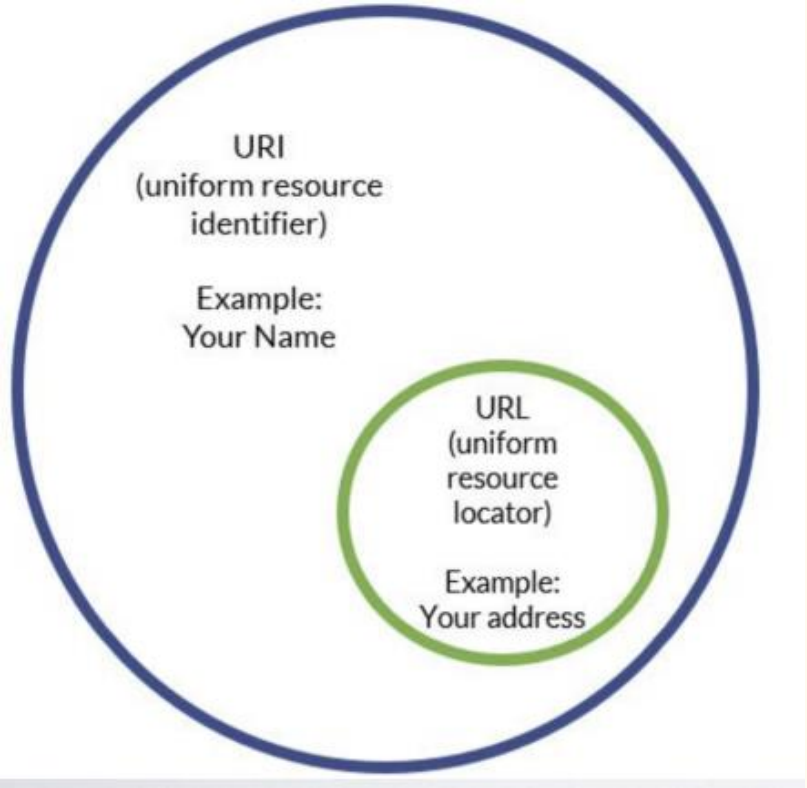
Look at an example

```
http://beginwithjava.com:80/servlet-jsp/introduction/url.html
|---| |----------------||--||-------------------------------|
  1         2              3           4
```

- 1 — Protocol name (HTTP)
- 2 — Host name
- 3 — Port number(Optional)
- 4 — Path

URL VS. URI

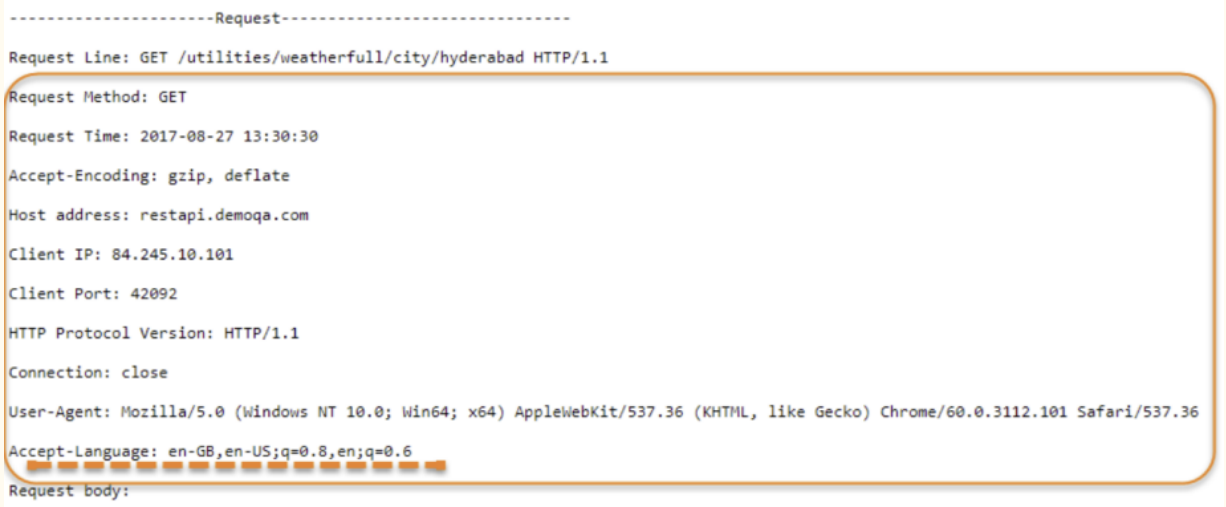
- URI can be a name, locator, or both for an online resource where a URL is just the locator
- URLs are a subset of URIs. That means all URLs are URIs.
- Your name could be a URI because it identifies you, but it couldn't be a URL because it doesn't help anyone find your location
- On the other hand, your address is both a URI and a URL because it both identifies you and it provides a location for you



Request Header

In the request section, whatever follows **Request Line** till before **Request Body** everything is a Header

Headers are used to pass additional information about the request to the server

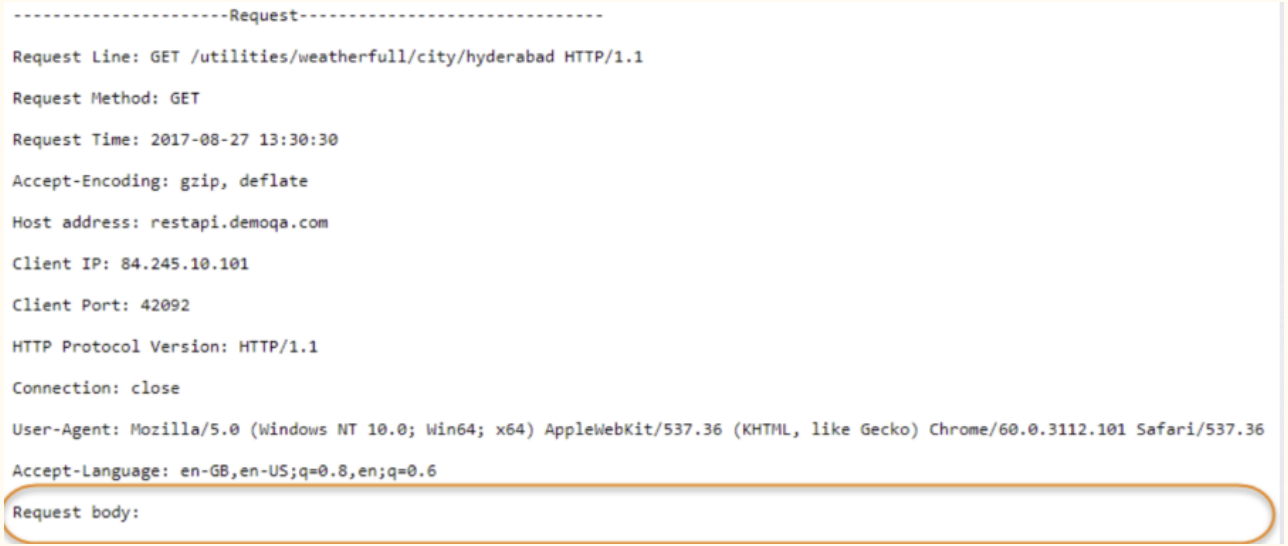


Request Body

Request Body is the part of the HTTP Request where additional content can be sent to the server.

- eg. a file type of JSON or XML.

It is optional



HTTP Response

HTTP Response is the packet of information sent by Server to the Client in response to an earlier Request made by Client

HTTP Response contains the information requested by the Client

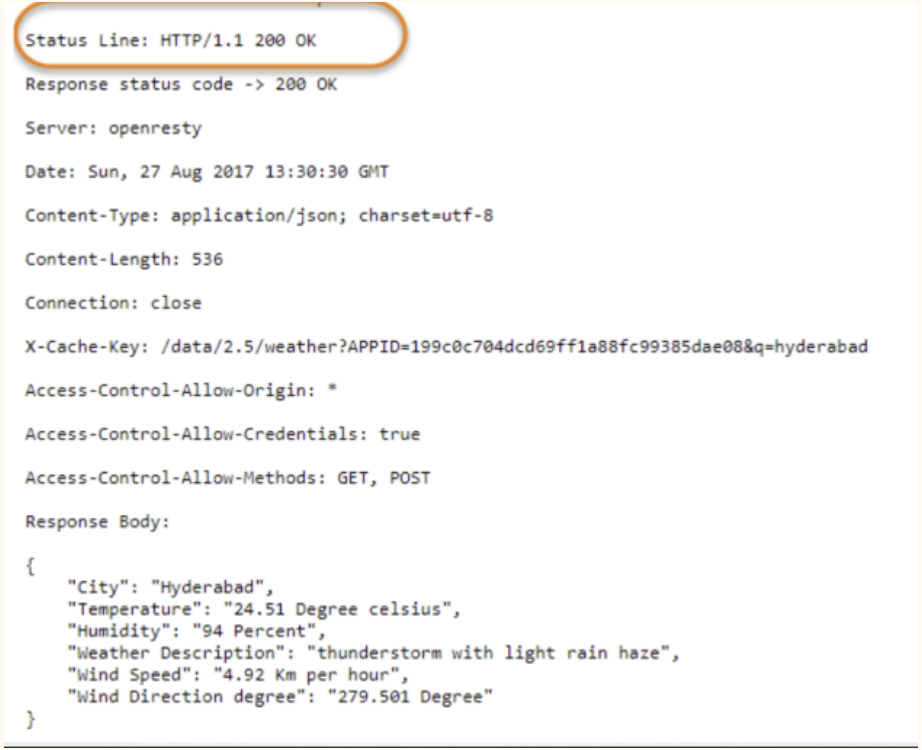
Just like HTTP Request, HTTP Response also has the same structure:

- Status Line
- Headers, 0 or more Headers in the request
- An optional Body of the Request

Status Line

A Status Line consists of three parts:

- HTTP Protocol Version
- Status Code
- Reason Phrase



HTTP Status Code

There are five classes of status code

- 1xx Informational – the request was received, continuing process
- 2xx Successful – the request was successfully received, understood and accepted
- 3xx Redirection – further action needs to be taken to complete the request
- 4xx Client Error – the request contains bad syntax or cannot be fulfilled
- 5xx Server Error – the server failed to fulfill an apparently valid request

Full status code reference: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Response Header

Just like a Request Header, Response Header also contains zero or more Header lines. However, it is very uncommon to have zero Headers in the response.

Lines just after the **Status Line** and before the **Response Body** are all Response Headers lines

Response Body

Response Body contains the resource data that was requested by the client.

Client

A **client** in web application usually refers to a server that presents the user interface which a user actually interacts with, like browsers.

Languages supported by web browsers:

- HTML
- CSS
- JavaScript

HTML

HTML stands for Hyper Text Markup Language

HTML describes the structure of a Web page

HTML consists of a series of elements

HTML elements tell the browser how to display the content

HTML elements are represented by tags

HTML tags label pieces of content such as "heading", "paragraph", "table", and so on

Browsers do not display the HTML tags, but use them to render the content of the page

CSS

CSS stands for Cascading Style Sheets

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS saves a lot of work. It can control the layout of multiple web pages all at once

External stylesheets are stored in CSS files

Javascript

JavaScript is the programming language of HTML and the Web.

- HTML to define the content of web pages
- CSS to specify the layout of web pages
- JavaScript to program the behavior of web pages

HTML and CSS will only produce static web pages. JavaScript will make it dynamic by different events to respond to different user interactions

Web Server

A program that uses HTTP for serving files that create web pages for users in response to their requests that are sent by the HTTP clients of their computer is called as a web server.

This server is always connected to the internet.

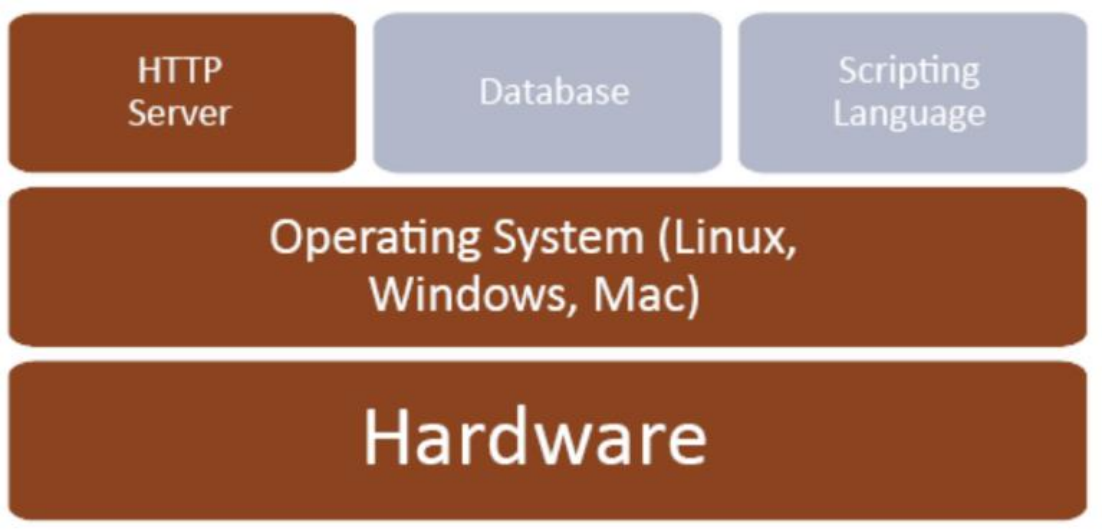
Every Web server that is connected to the Internet is given a unique address made up of a series of four numbers between 0 and 256 separated by periods.

- eg. 68.178.157.132

Web Server

Computers hosting websites are web servers

The diagram below represents the basic elements of a web server



Web Server

When you register a web address, also known as a domain name, such as google.com you have to specify the IP address of the Web server that will host the site

Then you can load up with Dedicated Servers that can support your web-based operations

There are four leading web servers

- Apache HTTP Server (like Tomcat) — widely used
- **Internet Information Services (IIS) — hosting for .NET**
- Nginx Web Server — high performance, stability, simple configuration and low resource usage.
- Google Web Server (GWS)

Except for those, there are several other web servers in market like Bea's Web Logic and IBM's WebSphere

Web Server

SMTP Server — Simple Mail Transfer Protocol Server. This server takes care of delivering emails from one server to another server

FTP — File Transfer Protocol. a standard network protocol used for the transfer of computer files between a client and server on a computer network

ISP — Internet Service Provider. They are the companies who provide you service in terms of internet connection to connect to the internet.

- eg. You will buy space on a Web Server from any Internet Service Provider. This space will be used to host your Website.

DNS — Domain Name System.

- When someone types in your domain name, www.example.com, your browser will ask the Domain Name System to find the IP that hosts your site. When you register your domain name, your IP address should be put in a DNS along with your domain name

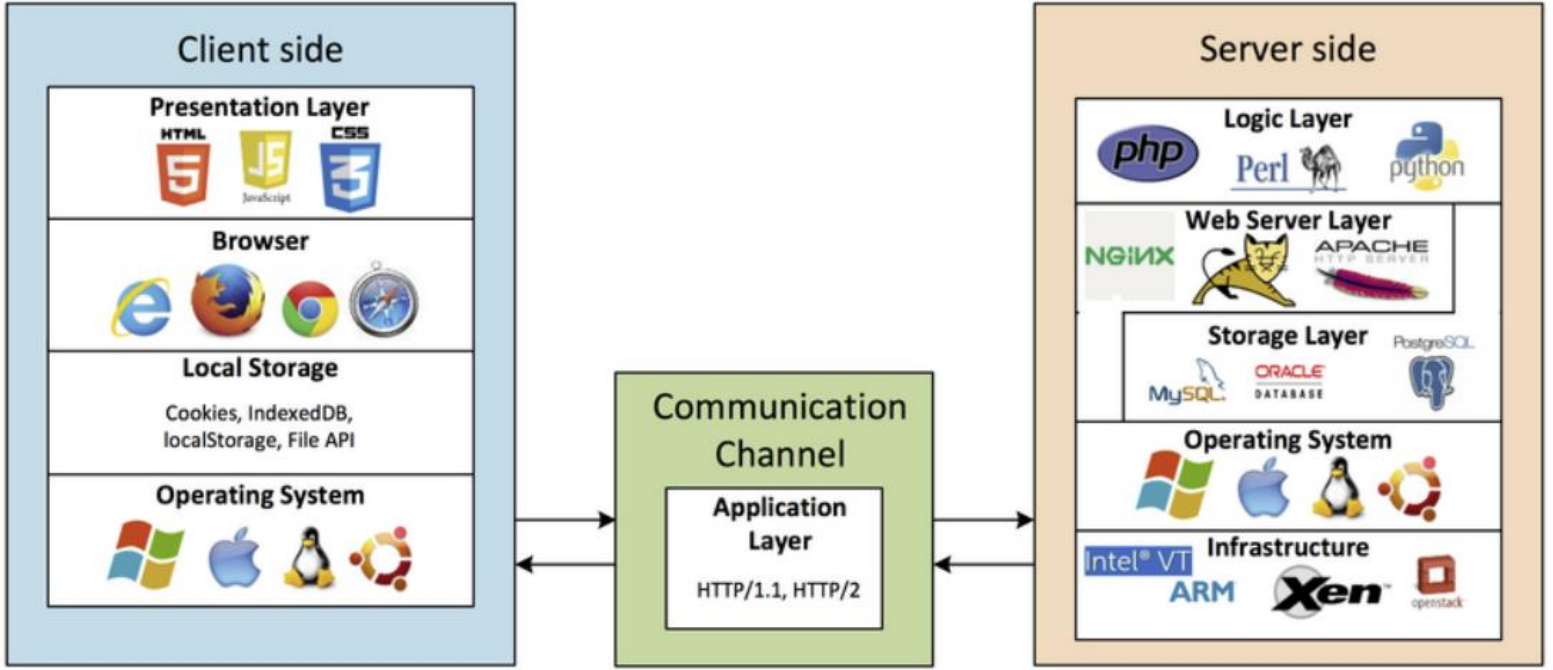
Web Application

- Until now, we have already covered all basics in web application.
- So what happens when I type **google.com** in the browser? Or how does the web application works?

How It Works

- A user enters a URL into a browser (for example, Google.com. This request is passed to a domain name server.
- The domain name server returns an IP address for the server that hosts the Website (for example, 68.178.157.132).
- The browser requests the page from the Web server using the IP address specified by the domain name server. (URL)
- The Web server returns the page to the IP address specified by the browser requesting the page. The page may also contain links to other files on the same server, such as images, which the browser will also request.
- The browser collects all the information and displays to your computer in the form of Web page.

Web Application

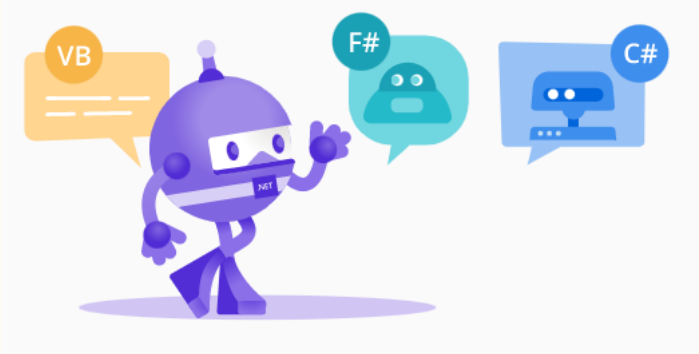


.NET Family

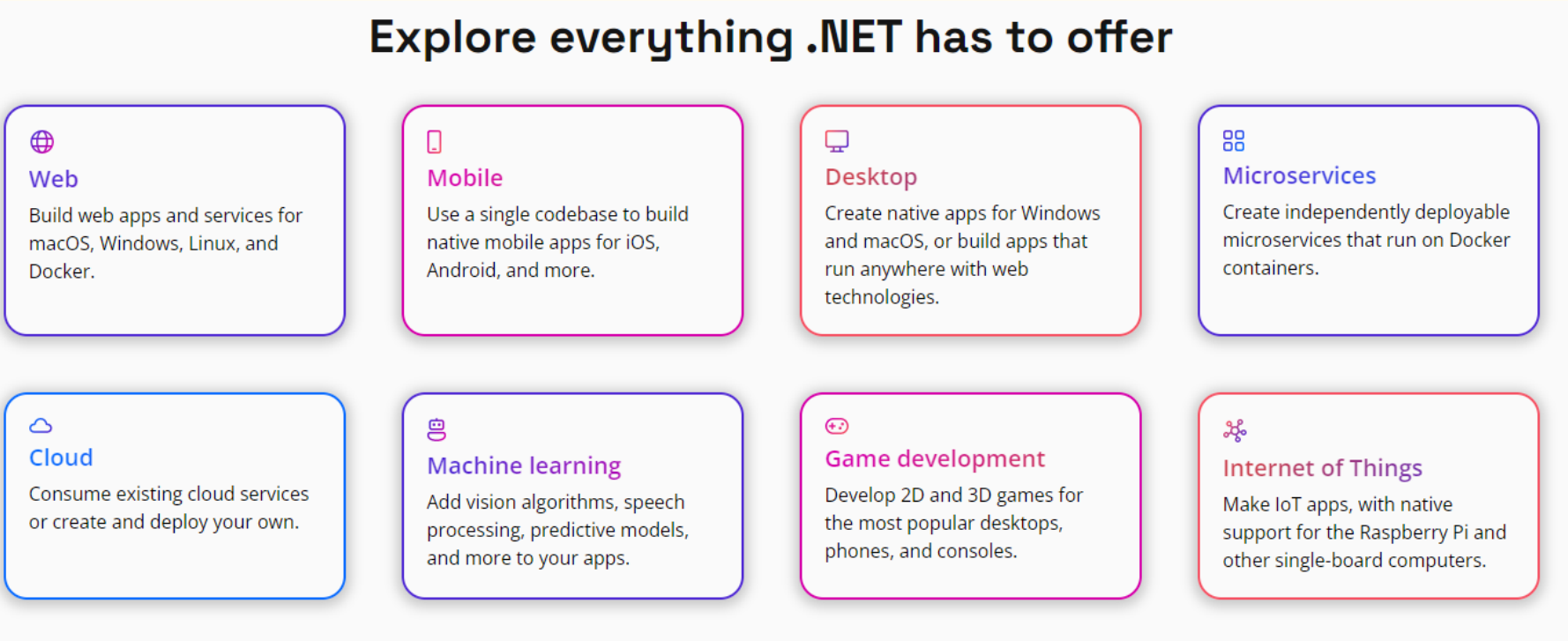
What is .NET

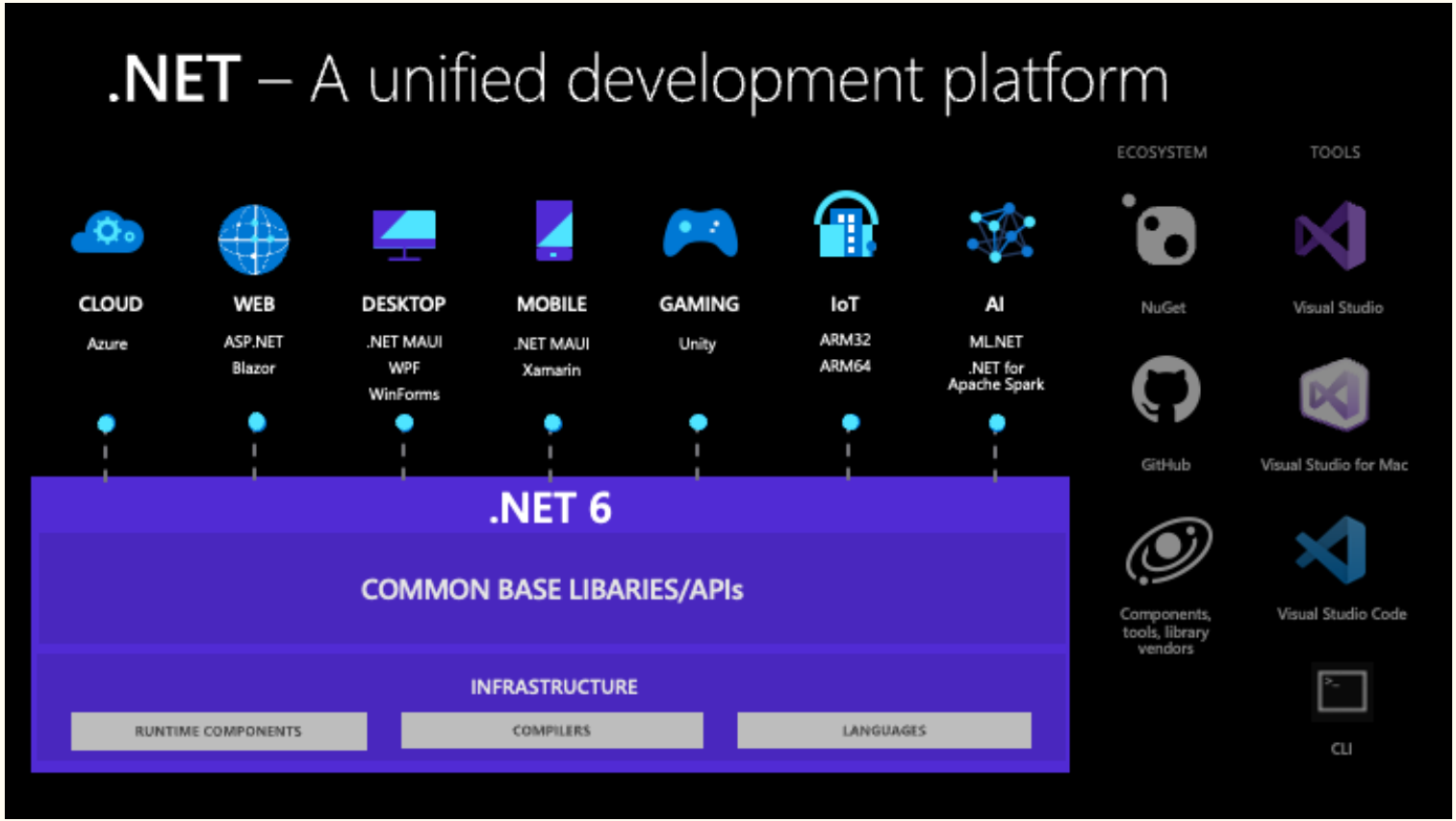
.NET is an ***open-source developer platform***, created by Microsoft, for building many different types of applications.

It's free and cross-platform, you can write .NET apps in C#, F# or Visual Basic.



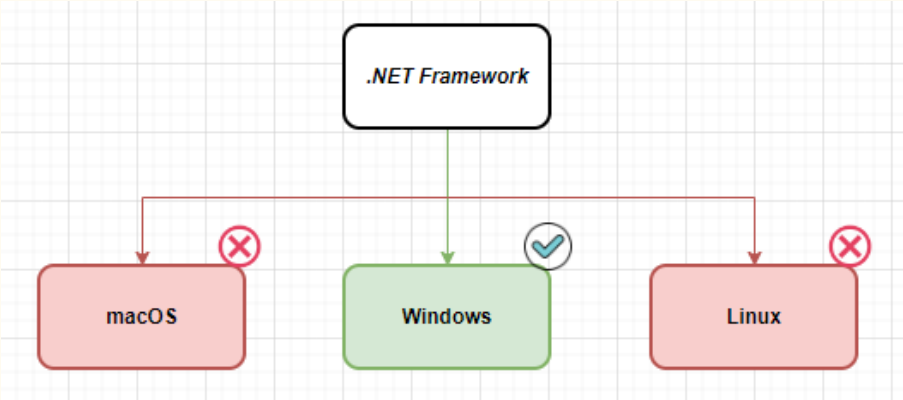
.NET Platform



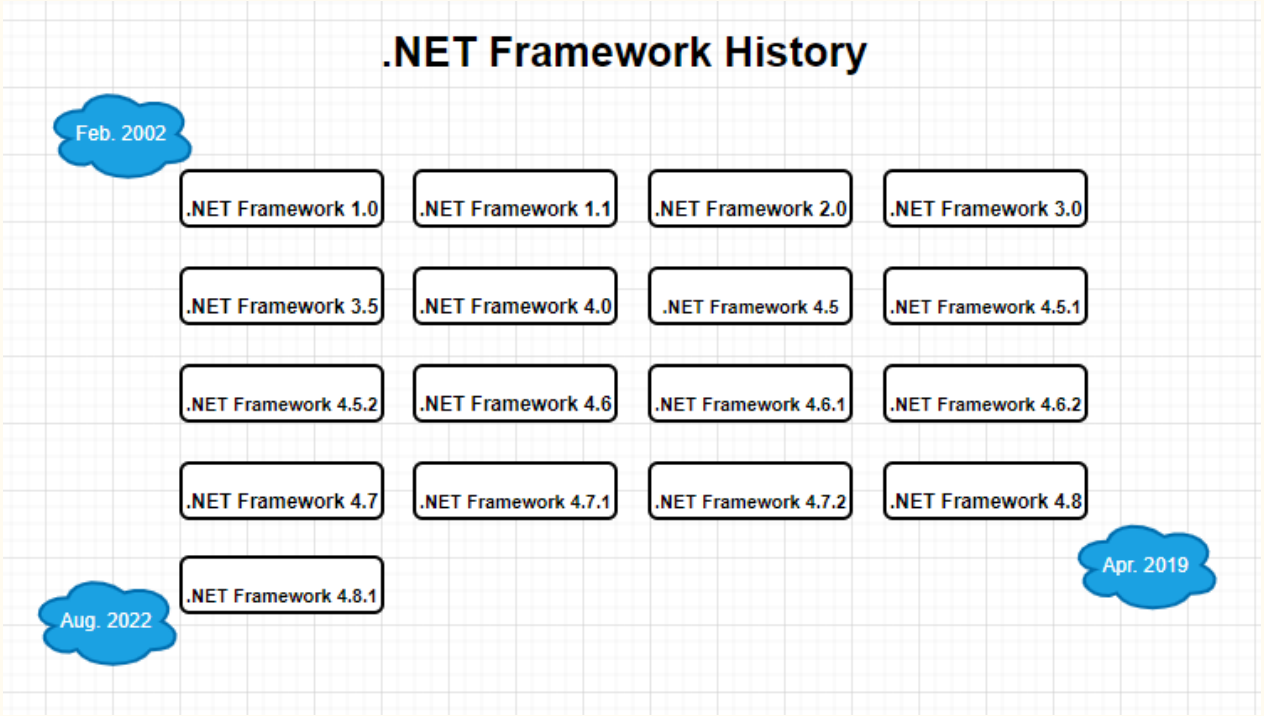

.NET – A unified development platform
CLOUD
Azure
WEB
ASP.NET
Blazor
DESKTOP
.NET MAUI
WPF
WinForms
MOBILE
.NET MAUI
Xamarin
GAMING
Unity
IoT
ARM32
ARM64
AI
ML.NET
.NET for Apache Spark
.NET 6
COMMON BASE LIBARIES/APIs
INFRASTRUCTURE
RUNTIME COMPONENTS
COMPILERS
LANGUAGES
ECOSYSTEM
TOOLS
NuGet
Visual Studio
GitHub
Visual Studio for Mac
Components, tools, library vendors
Visual Studio Code
CLI

.NET Framework

.NET Framework is the original implementation of .NET. It supports running websites, services, desktop apps, and more on **Windows.**

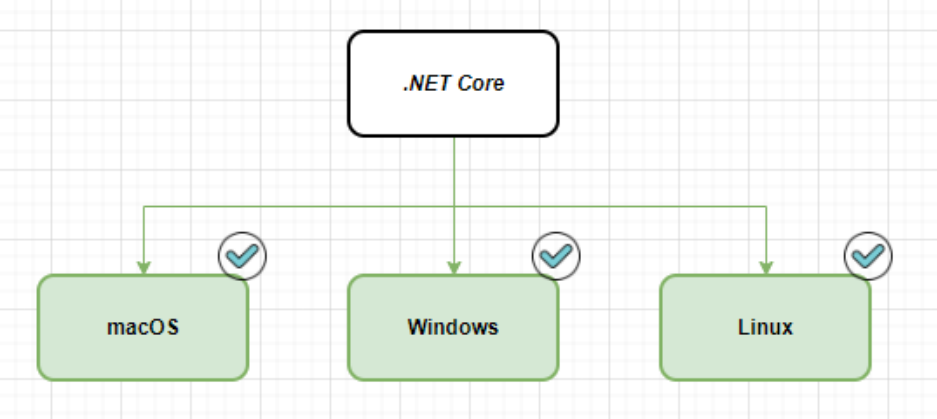


.NET Framework

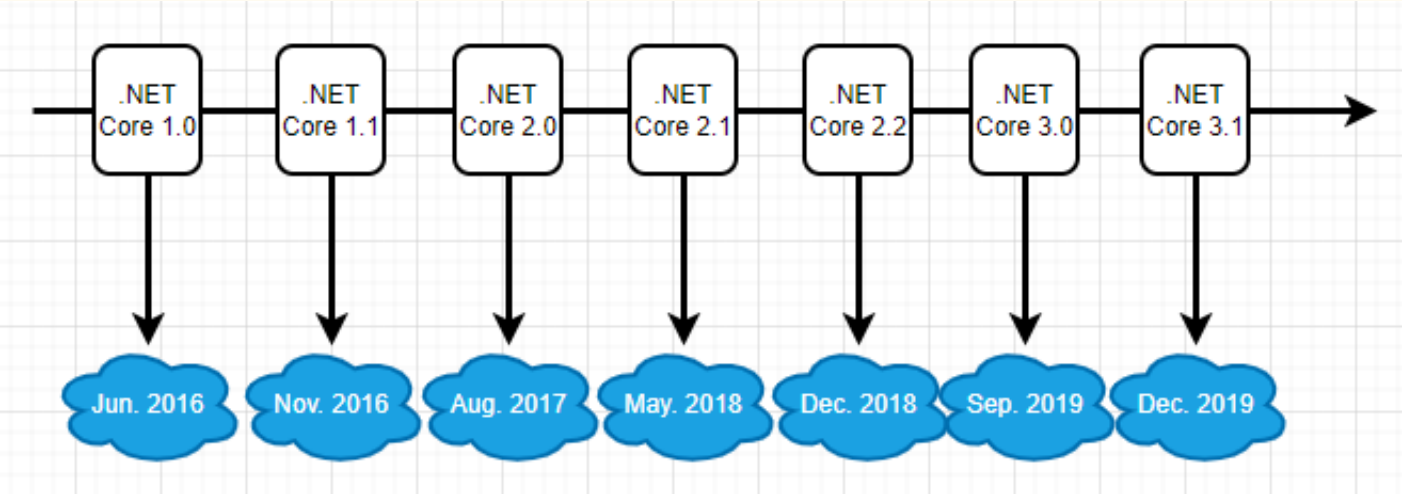


.NET Core

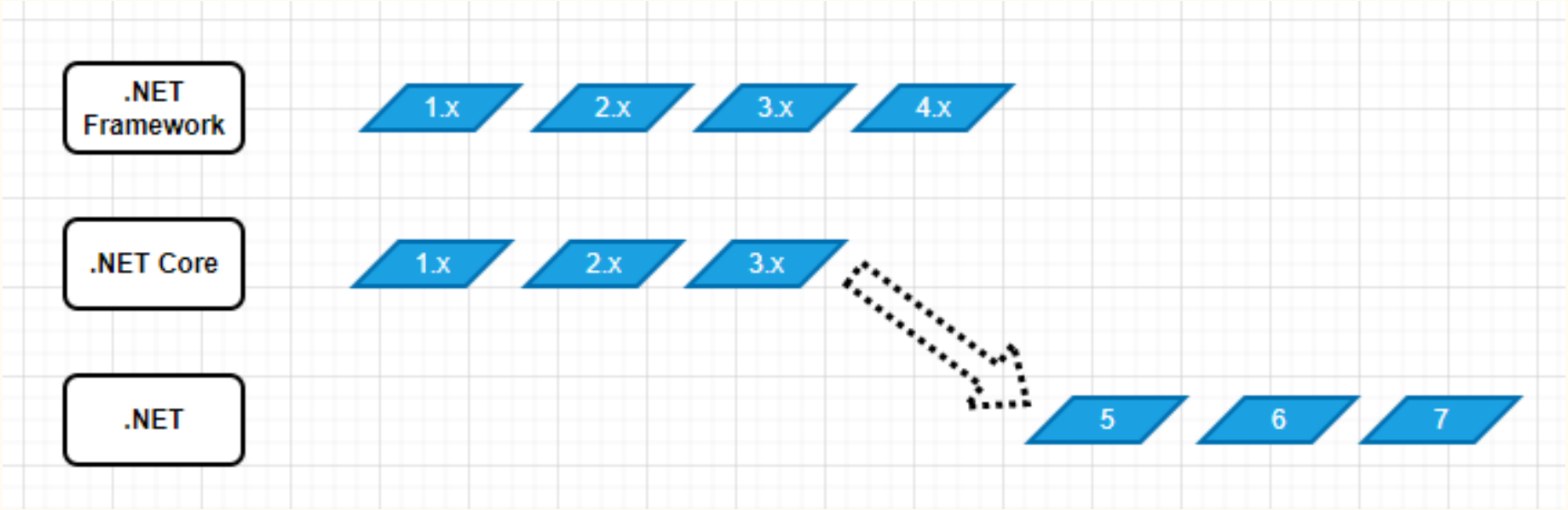
.NET Core is a free, open-source development platform maintained by Microsoft. It's a cross-platform that runs on Windows, macOS, and Linux operating systems.



.NET Core


.NET Core 1.0
.NET Core 1.1
.NET Core 2.0
.NET Core 2.1
.NET Core 2.2
.NET Core 3.0
.NET Core 3.1
Jun. 2016
Nov. 2016
Aug. 2017
May. 2018
Dec. 2018
Sep. 2019
Dec. 2019

.NET – One Unified Platform

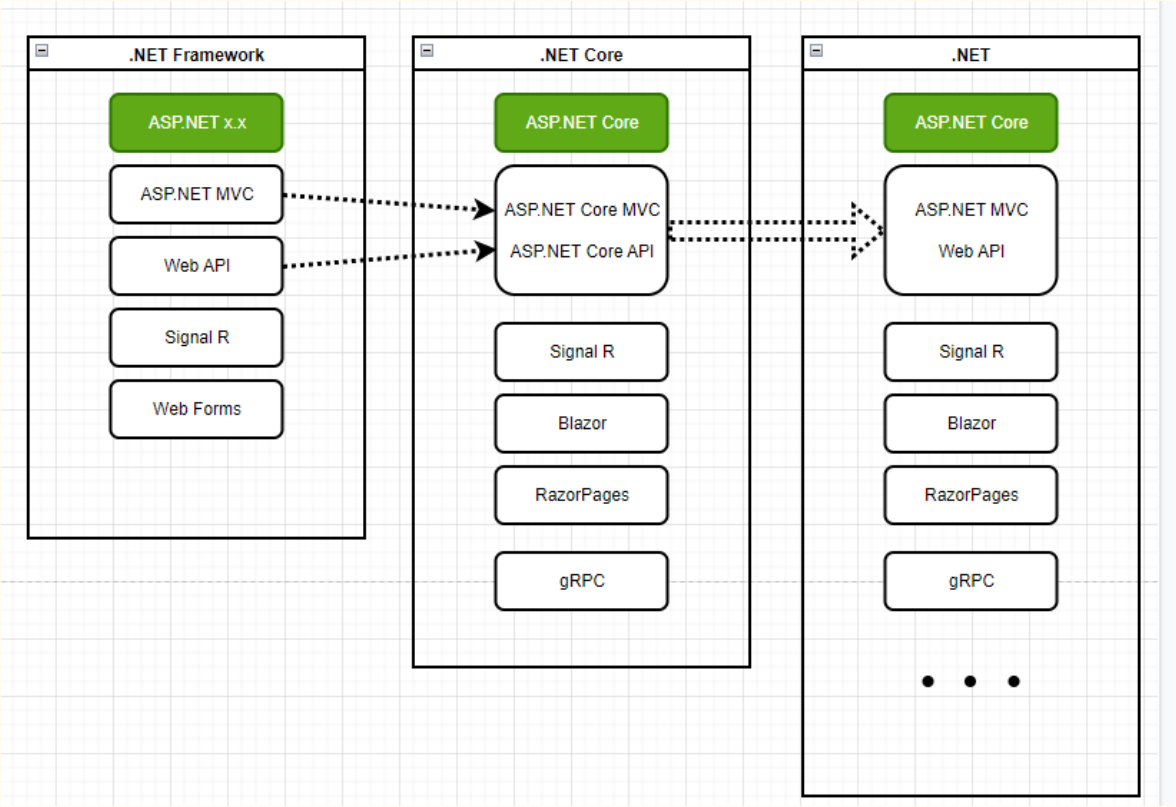


.NET and .NET Core supported versions

The following table tracks release and end of support dates for .NET and .NET Core versions.

Version	Original release date	Latest patch version	Patch release date	Release type	Support phase	End of support
.NET 7	November 8, 2022	7.0.0	November 8, 2022	STS	May 14, 2024	
.NET 6	November 8, 2021	6.0.11	November 8, 2022	LTS	Active	November 12, 2024
.NET Core 3.1	December 3, 2019	3.1.31	November 8, 2022	LTS	Maintenance	December 13, 2022

.NET vs. .NET Framework



ASP.NET vs. ASP.NET Core

ASP.NET Core	ASP.NET 4.x
Build for Windows, macOS, or Linux	Build for Windows
Razor Pages is the recommended approach to create a Web UI as of ASP.NET Core 2.x. See also MVC, Web API, and SignalR.	Use Web Forms, SignalR, MVC, Web API, WebHooks, or Web Pages
Multiple versions per machine	One version per machine
Develop with Visual Studio, Visual Studio for Mac, or Visual Studio Code using C# or F#	Develop with Visual Studio using C#, VB, or F#
Higher performance than ASP.NET 4.x	Good performance
Use .NET Core runtime	Use .NET Framework runtime

ASP.NET Core Project Type

- MVC
- API
- Razor Pages
- Blazor Server
- Blazor WebAssembly/Client

ASP.NET Core MVC

ASP.NET Core MVC provides features to build web APIs and web apps:

- The Model-View-Controller (MVC) pattern helps make your web APIs and web apps testable.
- Razor Pages is a page-based programming model that makes building web UI easier and more productive.
- Razor markup provides a productive syntax for Razor Pages and MVC views.
- Tag Helpers enable server-side code to participate in creating and rendering HTML elements in Razor files.
- Built-in support for multiple data formats and content negotiation lets your web APIs reach a broad range of clients, including browsers and mobile devices.
- Model binding automatically maps data from HTTP requests to action method parameters.
- Model validation automatically performs client-side and server-side validation.

Any Questions?